Appendix F

xSides Corporation
821 Second Avenue, #1600
Seattle, WA 98104

Pixel Bar API

The Application Programmer's Interface to the Pixel Bar

Table of Contents

Introduction

The Pixel Bar API is a set of C style functions that enable the xSides application to define pixel bars and display device dependent bitmaps in the pixel bars. The API includes functions to initialize the interface, create, modify, query and delete pixel bars, and to display colors or bitmaps in the pixel bars. For Win32 implementations, the API functions exist in a dynamically linked library (DLL).

The API functions include:

- **Pixelbar_Init** initializes and configures the API and the Pixel Bar software. The API functions are enabled.
- **Pixelbar_Uninit** restores the API and the Pixel Bar software to default undefined state. Other than Pixelbar_Init, all API functions are disabled.
- **Pixelbar_Create** creates a pixel bar and specifies its characteristics -- location, size, and whether it's visible or not. Also specifies the mode to use to acquire the screen area.
- **Pixelbar_CreateMR** creates a pixel bar and specifies its characteristics -- location, size, and whether it's visible or not for each supported display resolution. The mode used to acquire the screen area is also specified. This function differs from Pixelbar_Create in that when the display resolution changes, the xSides Space is automatically resized without further input from the xSides application.
- **Pixelbar_Modify** modifies the state of an existing pixel bar.
- **Pixelbar_Delete** deletes an existing pixel bar. The screen area is reclaimed, and the pixel bar is no longer displayed.
- **Pixelbar_Activate** makes a pixel bar active and visible on the screen.
- **Pixelbar_Deactivate** makes a pixel bar inactive and no longer visible on the screen. The screen area formerly occupied by the pixel bar is reclaimed.
- **Pixelbar_Query** returns the current state of the specified pixel bar, including location, size, visibility, etc.
- **Pixelbar_Enumerate** returns the number of pixel bars and their state.
- **Pixelbar_SetMouseEvent** sets a mouse event that will be reported to the application via a callback function.

- **Pixelbar_ClearMouseEvent** deletes the specified mouse event. The application will no longer receive reports for the mouse event.
- **Pixelbar_EnumerateMouseEvents** returns the list of current mouse events that the pixel bar software reports.
- **Pixelbar_QueryMouseEvent** returns the information of a specific mouse event the pixel bar software services.
- **Pixelbar_EnableMouse** enables the display of the mouse cursor icon in a pixel bar.
- **Pixelbar_DisableMouse** disables the display of the mouse cursor icon in a pixel bar.
- **Pixelbar_QueryDisplayInfo** returns the current physical screen resolution and other display information (e.g. refresh rate). The physical screen resolution doesn't include the screen areas occupied by pixel bars in overscan mode.
- **Pixelbar_DisplayColor** displays a color across the entire specified pixel bar. Used for testing.
- **Pixelbar_DisplayBitmap** displays a device dependent bitmap (i.e. a bitmap in the current screen format) at the specified location in the specified pixel bar.
- **Pixelbar_SetDisplayEvent** sets a display event that will be reported back to the application via a callback function.
- **Pixelbar_ClearDisplayEvent** deletes the specified display event.
- **Pixelbar_EnumerateDisplayEvents** returns the list of display events that are currently set.
- **Pixelbar_QuerySupport** returns the capabilities of the virtual xSides device.
- **Pixelbar_EnumerateFileInformation** returns the list of xSides driver files, their locations, names and versions.

A number of data structure types are used by the Pixel Bar API. They include:

- **PBConfiguration** describes the configuration of the pixel bar software for the current run.
- **PBPixelBarState** describes the current state of a pixel bar.
- **PBDisplayState** describes the current state of a display.
- **CURSOR_POS** contains the x and y coordinates of the cursor.

- **PBMouseEvent** contains the information that is passed back to the application's mouse event callback function in response to a mouse event.
- **PBMouseEventInfo** describes the current state of a mouse event set.
- **PBDisplayEvent** contains the information that is passed back to the application in response to a display event.

Pixel Bar API Functions

Pixelbar_Init

UINT32 Pixelbar_Init (PBConfiguration *lpPBConfiguration)

Initializes the pixel bar software and configures it using the data in the PBConfiguration parameter.

- Returns one of these values:

PB_SUCCESS	Function was successful.
PB_ERROR	An unrecoverable error occurred.

Parameters

LpPBConfiguration

Pointer to the configuration data structure. The pixel bar software is configured for the current session by the values in this data structure.

Notes

None.

Pixelbar_Uninit

UINT32 Pixelbar_Uninit(void)

Uninitializes the pixel bar software restores it to the uninitialized state. All resources allocated and used by the pixel bar software during an xSides session is freed.

- Returns one of these values:

PB_SUCCESS	Function was successful.
PB_ERROR	An unrecoverable error occurred.

Parameters

This function has no parameters.

Notes

None.

Pixelbar_Create

UINT32 Pixelbar_Create (UINT32 *lpPixelbarId, UINT32 xSidesMode, UINT32 Flags, UINT32 DisplayId, UINT32 Edge, INT32 OriginX, INT32 OriginY, UINT32 Width, UINT32 Height)

Pixelbar_Create() creates a pixel bar and returns the id. Memory is allocated for the pixel bar and, if created with the active flag, screen space is created on the display surface.

- Returns one of these values:

PB_SUCCESS	Function was successful.
PB_ERROR	An unrecoverable error occurred.

Parameters

lpPixelbarId

A pointer to the pixel bar id returned by Pixelbar_Create().

xSidesMode

The xSides mode that the pixel bar is to operate in. May be one of the following:

- MODE_SHARE Share mode -- shares the current display resolution with the desktop by decreasing the desktop size.
- MODE_STEPUP Step Up mode -- increases the physical resolution of the display, but keep the desktop at the current resolution and use the difference for displaying the pixel bar.
- MODE_STEPDOWN Step Down mode -- maintains the physical resolution of the display, but force the desktop to the next smaller defined resolution and use the difference for displaying the pixel bar.
- MODE_OVERSCAN Overscan mode -- reprograms the video hardware to provide extra scanlines at the bottom of the screen to display the pixel bar. This mode currently is valid only for pixel bars at the bottom or top edge of the display.
- MODE_AUTO Automatic mode -- the pixel bar software will attempt to use Overscan mode first, and if that's not possible, then use Share mode,

Flags

Flags specifies the options and features of the Pixelbar_Create() function and of the pixel bar that is created.

The options are specified by the following bitfields:

- PB_FLAGS_ACTIVE: Can be TRUE (1) or FALSE (0), which specify, respectively, whether the pixel bar is active or inactive upon creation. Active pixel bars are visible while inactive pixel bars are not visible, but may still be operated upon and updated.
- PB_FLAGS_EXACT_SIZE: Can be either TRUE (1) or FALSE (0), which specify respectively whether the pixel bar should be created with the exact size specified, returning an error if it's not possible, or with the closest size possible if there are not sufficient resources to provide the specified size. If the application creates the pixel bar with the PB_STATUS_EXACT_SIZE set to FALSE, it should query for the actual size created with Pixelbar_Query() function.
- PB_FLAGS_INVALID: What is this?
- PB_FLAGS_VALID_RES: What is this?
- PB_FLAGS_MOUSE_ENABLED: What is this?
- PB_FLAGS_DISPLAYED: What is this?

DisplayId

DisplayId specifies the display surface (aka monitor, screen) that the pixel bar will be displayed on. This feature supports multiple monitor systems. Currently, only DisplayId of 0 is supported.

Edge

Edge specifies which edge of the display surface the pixel bar will be displayed on. May be one of the following:

- PB_TOP_EDGE The top edge.
- PB_BOTTOM_EDGE The bottom edge.
- PB_RIGHT_EDGE The right edge. This edge is not valid if the pixel bar is in Overscan mode.
- PB_LEFT_EDGE The left edge. This edge is not valid if the pixel bar is in Overscan mode.

OriginX

OriginX specifies the x position on the display surface, in units of pixels with respect to the Desktop Origin, where the origin of the pixel bar (which is the upper left corner of the pixel bar) is located. The pixel bar origin must be consistent with its screen edge parameter; otherwise an error will be returned. This is a signed value and may be a negative value (which it will be for non-zero width pixel bars on the left edge of the screen.)

OriginY

OriginY specifies the y position on the display surface, in units of pixels with respect to the Desktop Origin, where the origin of the pixel bar (which is the upper left corner of the pixel bar) is located. The pixel bar origin must be consistent with its screen edge parameter; otherwise an error will be returned. This is a signed value and may be a negative value (which it will be for non-zero height pixel bars on the top edge of the screen.)

Width

Width specifies the desired width of the pixel bar, in number of pixels. It may not be larger than the total physical horizontal resolution of the display, which include the desktop and the pixel bars.

Height

Width specifies the desired height of the pixel bar, in number of pixels. It may not be larger than the total physical vertical resolution of the display, which include the desktop and the pixel bars.

Notes

The interaction of pixel bars running in different xSides modes is currently undefined, and therefore not supported. Hence, in a session, all pixel bars must be in the same xSides mode.

Pixelbar_CreateMR

UINT32 Pixelbar_CreateMR(UINT32 *lpPixelbarId, UINT32 xSidesMode, UINT32 Flags, UINT32 DisplayId, UINT32 Edge, PBPixelbarSize *lpPixelbarSize, UINT32 xSidesResolutionCount)

Pixelbar_CreateMR() creates a pixel bar and returns the id. This functions replaces Pixelbar_Create() and allows pixel bar sizes for various resolutions to be specified. This function differs from Pixelbar_Create in that when the display resolution changes, the xSides Space is automatically resized without further input from the xSides application.

- Returns one of these values:

PB_SUCCESS	Function was successful.
PB_ERROR	An unrecoverable error occurred.

Parameters

lpPixelbarId

A pointer to the pixel bar id returned by Pixelbar_CreateMR().

xSidesMode

The xSides mode that the pixel bar is to operate in. May be one of the following:

- MODE_SHARE Share mode -- shares the current display resolution with the desktop by decreasing the desktop size.
- MODE_STEPUP Step Up mode -- increases the physical resolution of the display, but keep the desktop at the current resolution and use the difference for displaying the pixel bar.
- MODE_STEPDOWN Step Down mode -- maintains the physical resolution of the display, but force the desktop to the next smaller defined resolution and use the difference for displaying the pixel bar.
- MODE_OVERSCAN Overscan mode -- reprograms the video hardware to provide extra scanlines at the bottom of the screen to display the pixel bar. This mode currently is valid only for pixel bars at the bottom or top edge of the display.
- MODE_AUTO Automatic mode -- the pixel bar software will attempt to use Overscan mode first, and if that's not possible, then use Share mode,

Flags

Flags specifies the options and features of the Pixelbar_Create() function and of the pixel bar that is created.

The options are specified by the following bitfields:

- PB_FLAGS_ACTIVE: Can be TRUE (1) or FALSE (0), which specify, respectively, whether the pixel bar is active or inactive upon creation. Active pixel bars are visible while inactive pixel bars are not visible, but may still be operated upon and updated.
- PB_FLAGS_EXACT_SIZE: Can be either TRUE (1) or FALSE (0), which specify respectively whether the pixel bar should be created with the exact size specified, returning an error if it's not possible, or with the closest size possible if there are not sufficient resources to provide the specified size. If the application creates the pixel bar with the PB_STATUS_EXACT_SIZE set to FALSE, it should query for the actual size created with Pixelbar_Query() function.
- PB_FLAGS_INVALID: What is this?
- PB_FLAGS_VALID_RES: What is this?
- PB_FLAGS_MOUSE_ENABLED: What is this?
- PB_FLAGS_DISPLAYED: What is this?

DisplayId

DisplayId specifies the display surface (aka monitor, screen) that the pixel bar will be displayed on. This feature supports multiple monitor systems. Currently, only DisplayId of 0 is supported.

Edge

Edge specifies which edge of the display surface the pixel bar will be displayed on. May be one of the following:

PB_TOP_EDGE	The top edge.
PB_BOTTOM_EDGE	The bottom edge.
PB_RIGHT_EDGE	The right edge. This edge is not valid if the pixel bar is in Overscan mode.
PB_LEFT_EDGE	The left edge. This edge is not valid if the pixel bar is in Overscan mode.

lpPixelbarSize

lpPixelbarState is a pointer to an array of pixel bar size and origin data for various screen resolutions. The number of entries in the array is specified by xSidesResolutionCount.

xSidesResolutionCount

xSidesResolutionCount indicates the number of xSides resolutions that are listed by lpPixelbarSize.

Notes

The interaction of pixel bars running in different xSides modes is currently undefined, and therefore not supported. Hence, in a session, all pixel bars must be in the same xSides mode.

Pixelbar_Modify

UINT32 Pixelbar_Modify (UINT32 PixelbarId, UINT32 Flags, INT32 OriginX, INT32 OriginY, UINT32 Width, UINT32 Height)

Pixelbar_Modify() modifies the size and position of an existing pixel bar. The xSides mode and the edge are not affected.

- Returns one of these values:

PB_SUCCESS	Function was successful.
PB_ERROR	An unrecoverable error occurred.

Parameters

PixelbarId

The id of the pixel bar.

Status

Status specifies the options and features of the Pixelbar_Modify() function and of the pixel bar.

The options are specified by the following bitfields:

PB_STATUS_ACTIVE: Can be TRUE (1) or FALSE (0), which specify, respectively, whether the pixel bar is active or inactive upon creation. Active pixel bars are visible while inactive pixel bars are not visible, but may still be operated upon and updated.

PB_STATUS_EXACT_SIZE: Can be either TRUE (1) or FALSE (0), which specify respectively whether the pixel bar should be created with the exact size specified, returning an error if it's not possible, or with the closest size possible if there are not sufficient resources to provide the specified size. If the application creates the pixel bar with the PB_STATUS_EXACT_SIZE set to FALSE, it should query for the actual size created with Pixelbar_Query() function.

OriginX

OriginX specifies the x position on the display surface, in units of pixels with respect to the Desktop Origin, where the origin of the pixel bar (which is the upper left corner of the pixel bar) is located. The pixel bar origin must be consistent with its screen edge parameter; otherwise an error will be returned. This is a signed value and may be a negative value (which it will be for non-zero width pixel bars on the left edge of the screen.)

OriginY

OriginY specifies the y position on the display surface, in units of pixels with respect to the Desktop Origin, where the origin of the pixel bar (which is the upper left corner of the pixel bar) is located. The pixel bar origin must be consistent with its screen edge parameter; otherwise an error will be returned. This is a signed value and may be a negative value (which it will be for non-zero height pixel bars on the top edge of the screen.)

Width

Width specifies the desired width of the pixel bar, in number of pixels. It may not be larger than the total physical horizontal resolution of the display, which include the desktop and the pixel bars.

Height

Width specifies the desired height of the pixel bar, in number of pixels. It may not be larger than the total physical vertical resolution of the display, which include the desktop and the pixel bars.

Notes

Pixelbar_Modify is currently not supported. As currently defined, it is unable to modify the origins and sizes of pixel bars created with the Pixelbar_CreateMR() function. Pixelbar_Modify potentially may be removed or redefined.

Pixelbar_Delete

UINT32 Pixelbar_Delete (UINT32 PixelbarId)

Pixelbar_Delete() deletes an existing pixel bar and frees its resources.

- Returns one of these values:

PB_SUCCESS	Function was successful.
PB_ERROR	An unrecoverable error occurred.

Parameters

PixelbarId

The id of the pixel bar.

Notes

None.

Pixelbar_Activate

UINT32 Pixelbar_Activate (UINT32 PixelbarId)

Pixelbar_Activate() activates an existing pixel bar and makes it visible. If the pixel bar uses MODE_SHARE, the desktop is reduced and space on the display surface is created for the pixel bar. . If the pixel bar uses MODE_OVERSCAN, the video hardware is reprogrammed to create the scan lines used by the pixel bar

- Returns one of these values:

PB_SUCCESS	Function was successful.
PB_ERROR	An unrecoverable error occurred.

Parameters

PixelbarId

The id of the pixel bar.

Notes

None.

Pixelbar_Deactivate

UINT32 Pixelbar_Deactivate (UINT32 PixelbarId)

Pixelbar_Deactivate() deactivates an existing pixel bar and makes it no longer visible. If the pixel bar uses MODE_SHARE, then the screen space is returned to the desktop. If the pixel bar uses MODE_OVERSCAN, the video hardware is reprogrammed to remove the scan lines used by the pixel bar.

- Returns one of these values:

PB_SUCCESS	Function was successful.
PB_ERROR	An unrecoverable error occurred.

Parameters

PixelbarId

The id of the pixel bar.

Notes

The behavior Pixelbar_Deativate() if the pixel bar uses MODE_STEPUP or MODE_STEPDOWN is currently neither defined nor supported..

Pixelbar_Enumerate

UINT32 Pixelbar_Enumerate (UINT32 *lpPixelbarCount, UINT32 **lpPixelbarIdTable, UINT32 *lpPixelbarsReturned, UINT32 *lpszData, UINT32 szOutputBuffer)

Pixelbar_Enumerate() returns the count of existing pixel bars and their ids. The application can then get the characteristics of the individual pixel bars by calling Pixelbar_Query().

- Returns one of these values:

PB_SUCCESS	Function was successful.
PB_ERROR	An unrecoverable error occurred.

PB_BUFFER_TOO_SMALL	The provided output buffer is too small to contain all the requested information. The information returned in the output buffer is not guaranteed to be coherent or valid. The function should be called again with an output buffer of the appropriate size.

Parameters

lpPixelbarCount

lpPixelbarCount is the address of an UINT32 where the count of existing pixels will be returned.

lpPixelbarIdTable

lpPixelbarIdTable is the address of an array of pixel bar ids where the existing pixel bar ids will be returned.

lpPixelbarsReturned

lpPixelbarsReturned is the address of an UINT32 where the count of pixels whose ids are returned in lpPixelbarIdTable will be written. Unless the output buffer pointed to by lpPixelbarIdTable is too small; the value in lpPixelbarsReturned is the same as in lpPixelbarCount.

lpszData

The address of the size of the requested data. If the value is larger than szOutputBuffer, then the provided buffer is too small, and the error code PB_BUFFER_TOO_SMALL is returned by the function.

szOutputBuffer

The size of the output buffer pointed to by lpPixelbarIdtable, in bytes. This will allow the Pixel Bar software to refrain from overflowing the buffer.

Notes

None.

Pixelbar_Query

UINT32 Pixelbar_Query (UINT32 PixelbarId, PBPixelbarState *lpPBState, UINT32 *lpszData, UINT32 szOutputBuffer)

Pixelbar_Query() returns the state and characteristics of the specified pixel bar, including whether it is active, its width and height, etc.

- Returns one of these values:

PB_SUCCESS	Function was successful.
PB_ERROR	An unrecoverable error occurred.
PB_BUFFER_TOO_SMALL	The provided output buffer is too small to contain all the requested information. The information returned in the output buffer is not guaranteed to be coherent or valid. The function should be called again with an output buffer of the appropriate size.

Parameters

Pixelbar_Id

The id of the pixel bar.

lpPBState

The address of a PBPixelbarState structure that the pixel bar state is returned in.

lpszData

The address of the size of the requested data. If the value is larger than szOutputBuffer, then the provided buffer is too small, and the error code PB_BUFFER_TOO_SMALL is returned by the function.

szOutputBuffer

The size of the output buffer pointed to by lpPBState, in bytes. This will allow the Pixel Bar software to refrain from overflowing the buffer.

Notes

None.

Pixelbar_SetMouseEvent

UINT32 Pixelbar_SetMouseEvent (UINT32 PixelbarId, UINT32 *lpMouseEventId, UINT32 MouseEvent, UINT32 (*fpCallback)(PBMouseEvent *lpParam))

Pixelbar_SetMouseEvent provides the application with a method to be notified of mouse events. Multiple mouse events can be requested with a single call to Pixelbar_SetMouseEvent; an Identifier is returned for each request. When multiple requests

specify the same mouse events (e.g. PB_MOUSE_MOVEMENT), all requests are serviced in the TBD [*possibly reverse*] order that they were requested. The notification is performed via function callbacks.

- Returns one of these values:

PB_SUCCESS	Function was successful.
PB_ERROR	An unrecoverable error occurred.

Parameters

Pixelbar_Id

The id of the pixel bar.

lpMouseEventId

The address of a UINT32 that in which the Pixel Bar software returns the Mouse Event Id.

MouseEvent

The mouse event that triggers the callback function. It may be one or more of the following:

PB_MOUSE_LBUTTON_DOWN	Trigger when the cursor is in the pixel bar and the left mouse button is depressed.
PB_MOUSE_LBUTTON_UP	Trigger when the cursor is in the pixel bar and the left mouse button is released.
PB_MOUSE_RBUTTON_DOWN	Trigger when the cursor is in the pixel bar and the right mouse button is depressed.
PB_MOUSE_RBUTTON_UP	Trigger when the cursor is in the pixel bar and the right mouse button is released.
PB_MOUSE_MBUTTON_DOWN	Trigger when the cursor is in the pixel bar and the middle mouse button is depressed.
PB_MOUSE_MBUTTON_UP	Trigger when the cursor is in the pixel bar and the middle mouse button is released.
PB_MOUSE_LDBL_CLICK	Trigger when the cursor is in the pixel bar and there is a double click on the left button.

PB_MOUSE_RDBL_CLICK	Trigger when the cursor is in the pixel bar and there is a double click on the right button.
PB_MOUSE_MDBL_CLICK	Trigger when the cursor is in the pixel bar and there is a double click on the middle button.
PB_MOUSE_MOVEMENT	Trigger when the cursor is in the pixel bar and its position changes.
PB_MOUSE_LEAVE	Trigger when the cursor leaves the pixel bar.

fpCallback

The function to call when the specified mouse event occurs. The function returns a status and has one parameter that is a pointer to the mouse event data structure.

Returns

PB_SUCCESS	Function was successful.
PB_ERROR	An unrecoverable error occurred.

lpParam

lpParam is a pointer to the PBMouseEvent data structure. The pixel bar, the mouse event, and if applicable, the x and y locations of the mouse event are reported.

Notes

The issue of mouse interactions is being researched.

Pixelbar_ClearMouseEvent

UINT32 Pixelbar_ClearMouseEvent (UINT32 PixelbarId, UINT32 MouseEventId)

Pixelbar_ClearMouseEvent() clears a previously requested set of mouse events. When the mouse event set is cleared, the Pixel Bar software will no longer service the set.

- Returns one of these values:

PB_SUCCESS	Function was successful.

PB_ERROR	An unrecoverable error occurred.

Parameters

Pixelbar_Id

The id of the pixel bar.

MouseEventId

The Id of the mouse event set. This value was returned by a previous call to Pixelbar_SetMouseEvent.

Notes

The issue of mouse interactions is being researched.

Pixelbar_EnumerateMouseEvents

UINT32 Pixelbar_EnumerateMouseEvents (UINT32 PixelbarId, UINT32 *lpMouseEventIdCount, UINT32 **lpMouseEventIds, UINT32 *lpMouseEventIdsReturned, UINT32 *lpszData, UINT32 szOutputBuffer)

Pixelbar_EnumerateMouseEvents() returns the number of mouse events and the list of mouse events that have been set for the specified pixel bar.

- Returns one of these values:

PB_SUCCESS	Function was successful.
PB_ERROR	An unrecoverable error occurred.
PB_BUFFER_TOO_SMALL	The provided output buffer is too small to contain all the requested information. The information returned in the output buffer is not guaranteed to be coherent or valid. The function should be called again with an output buffer of the appropriate size.

Parameters

PixelbarId

The id of the pixel bar.

lpMouseEventIdCount

The address of the UINT32 where the count of mouse events will be returned.

lpMouseEventIds

The address of an array of UINT32 that hold the Ids of Mouse Event sets.

lpMouseEventIdsReturned

lpMouseEventsReturned is the address of an UINT32 where the count of mouse event ids, which are returned in lpMouseEventIds, will be written. Unless the output buffer pointed to by lpMouseEventIds is too small; the value in lpMouseEventIdsReturned is the same as in lpMouseEventIdCount.

lpszData

The address of the size of the requested data. If the value is larger than szOutputBuffer, then the provided buffer is too small, and the error code PB_BUFFER_TOO_SMALL is returned by the function.

szOutputBuffer

The size of the output buffer pointed to by lpMouseEventIds, in bytes. This will allow the Pixel Bar software to refrain from overflowing the buffer.

Notes

None.

Pixelbar_QueryMouseEvent

UINT32 Pixelbar_QueryMouseEvent (UINT32 PixelbarId, UINT32 MouseEventId, PBMouseEventInfo *lpMouseEventInfo, UINT32 *lpszData, UINT32 szOutputBuffer)

Pixelbar_QueryMouseEvent() returns the mouse event state for the specified Mouse Event ID.

- Returns one of these values:

PB_SUCCESS	Function was successful.
PB_ERROR	An unrecoverable error occurred.
PB_BUFFER_TOO_SMALL	The provided output buffer is too small to contain all the requested information. The information returned in the output buffer is not guaranteed to be coherent or valid. The function should be called again with an output buffer of the appropriate size.

Parameters

PixelbarId

The id of the pixel bar.

MouseEventId

The Id of the Mouse Event instance.

lpMouseEventInfo

The address of a PBMouseEventInfo structure that the mouse event information is returned in.

lpszData

The address of the size of the requested data. If the value is larger than szOutputBuffer, then the provided buffer is too small, and the error code PB_BUFFER_TOO_SMALL is returned by the function.

szOutputBuffer

The size of the output buffer pointed to by lpMouseEventInfo, in bytes. This will allow the Pixel Bar software to refrain from overflowing the buffer.

Notes

The issue of mouse interactions is being researched.

Pixelbar_EnableMouse

UINT32 Pixelbar_EnableMouse (UINT32 PixelbarId)

Pixelbar_EnableMouse enables the display of the mouse cursor icon in the specified pixel bar.

Returns one of these values:

PB_SUCCESS	Function was successful.
PB_ERROR	An unrecoverable error occurred.

Parameters

Pixelbar_Id

The id of the pixel bar.

Notes

None.

Pixelbar_DisableMouse

UINT32 Pixelbar_DisableMouse (UINT32 PixelbarId)

Pixelbar_DisableMouse disables the display of the mouse cursor icon in the specified pixel bar.

Returns one of these values:

PB_SUCCESS	Function was successful.
PB_ERROR	An unrecoverable error occurred.

Parameters

Pixelbar_Id

The id of the pixel bar.

Notes

None.

Pixelbar_QueryDisplayInfo

UINT32 Pixelbar_QueryDisplayInfo (UINT32 DisplayId, PBDisplayInfo *lpDisplayInfo, UINT32 *lpszData, UINT32 szOutputBuffer)

Pixelbar_QueryDisplayInfo() queries the pixel bar software for the characteristics and current state of the specified display.

- Returns one of these values:

PB_SUCCESS	Function was successful.
PB_ERROR	An unrecoverable error occurred.
PB_BUFFER_TOO_SMALL	The provided output buffer is too small to contain all the requested information. The information returned in the output buffer is not guaranteed to be coherent or valid. The function should be called again with an output buffer of the appropriate size.

Parameters

DisplayId

The display to query.

lpDisplayInfo

The address of the structure where the display info is returned.

lpszData

The address of the size of the requested data. If the value is larger than szOutputBuffer, then the provided buffer is too small, and the error code PB_BUFFER_TOO_SMALL is returned by the function.

szOutputBuffer

The size of the output buffer pointed to by lpDisplayInfo, in bytes. This will allow the Pixel Bar software to refrain from overflowing the buffer.

Notes

The physical resolution doesn't include the scanlines used by pixel bars in MODE_OVERSCAN.

Pixelbar_DisplayColor

UINT32 Pixelbar_DisplayColor (UINT32 PixelbarId, UINT32 DestX, UINT32 DestY, UINT32 Xext, UINT32 Yext, UINT32 Color)

Pixelbar_DisplayColor() specifies a single color value in the current display format that the specified rectangle in the pixel bar is to be colored with.

- Returns one of these values:

PB_SUCCESS	Function was successful.
PB_ERROR	An unrecoverable error occurred.

Parameters

PixelbarId

The pixel bar to color.

DestX

The x origin, with respect, with respect to the pixel bar origin, of the rectangle in the pixel bar that is to be colored. This value must be within the bounds of the destination pixel bar

DestY

The x origin, with respect, with respect to the pixel bar origin, of the rectangle in the pixel bar that is to be colored. This value must be within the bounds of the destination pixel bar

Xext

The width of the rectangle to be colored. The sum of DestX and Xext must be less than the width of the destination pixel bar.

Yext

The height of the rectangle to be colored. The sum of DestY and Yext must be less than the height of the destination pixel bar

Color

A single pixel's worth of color value to color the specified rectangle within the pixel bar. The color is in the same format as the screen. For 8 bits per pixel resolutions, only the least significant 8 bits have valid data; for 16 bits per pixel resolutions, only the least significant 16 bits have valid data; and for 24 bits per pixel resolutions, only the least significant 24 bits have valid data. The remaining bits should be 0 (zero). For 32 bits per pixel resolutions, the most significant 8 bits may be set to an Alpha value, which may or may not affect the displayed image, depending on the video device.

For palettized mode displays (e.g. 8 bits per pixel), the pixel bar software currently does not provide a facility to load the color palette, and thus whatever color palette is loaded in the system will be used.

Notes

The pixel bar coordinate system is the same kind as that of the display, with a different origin. For Windows pixel bars, the origin is at the upper left corner of the pixel bar, and the x values increase from left to right, and the y values increase from top to bottom.

Pixelbar_DisplayBitmap

UINT32 Pixelbar_DisplayBitmap (UINT32 PixelbarId, UINT32 SrcX, UINT32 SrcY, UINT32 DestX, UINT32 DestY, UINT32 Xext, UINT32 Yext, UINT32 szBitmap, UINT32 SrcStride, VOID *lpBitmap)

Pixelbar_DisplayBitmap() displays the specified bitmap onto the specified rectangle in the pixel bar.

- Returns one of these values:

PB_SUCCESS	Function was successful.
PB_ERROR	An unrecoverable error occurred.

Parameters

PixelbarId

The destination pixel bar.

SrcX

The x origin, with respect, with respect to the source bitmap origin, of the source rectangle in the source bitmap. The origin is at the upper left corner of the bitmap. This value must be within the bounds of the source bitmap.

SrcY

The y origin, with respect, with respect to the source bitmap origin, of the source rectangle in the source bitmap. The origin is at the upper left corner of the bitmap. This value must be within the bounds of the source bitmap.

DestX

The x origin, with respect, with respect to the pixel bar origin, of the destination rectangle. This value must be within the bounds of the destination pixel bar.

DestY

The x origin, with respect, with respect to the pixel bar origin, of the destination rectangle. This value must be within the bounds of the destination pixel bar.

Xext

The width of the destination rectangle. The sum of SrcX and Xext must be less than the width of the source bitmap. Also the sum of DestX and Xext must be less than the width of the destination pixel bar.

Yext

The height of the destination rectangle. The sum of SrcY and Yext must be less than the height of the source bitmap. Also the sum of DestY and Yext must be less than the height of the destination pixel bar.

szBitmap

The size (in the number of bytes) of the source bitmap.

SrcStride

The number of bytes in each horizontal line of the source bitmap.

lpBitmap

The address of the source bitmap. The bitmap must be in the same screen format (e.g. number of bits per pixel, bits per color, etc.) as the destination display.

Notes

The pixel bar coordinate system is the same kind as that of the display, with a different origin. For Windows pixel bars, the origin is at the upper left corner of the pixel bar, and the x values increase from left to right, and the y values increase from top to bottom.

For palettized mode displays (e.g. 8 bits per pixel), the pixel bar software currently does not provide a facility to load the color palette, and thus whatever color palette is loaded in the system will be used.

Pixelbar_SetDisplayEvent

UINT32 Pixelbar_SetDisplayEvent (UINT32 DisplayId, UINT32 DisplayEvent, UINT32 (*fpCallback)(PBDisplayEvent *lpParam))

Pixelbar_SetDisplayEvent provides the application with a method to be asynchronously notified of various xSides events. The notification is performed via function callbacks.

- Returns one of these values:

PB_SUCCESS	Function was successful.
PB_ERROR	An unrecoverable error occurred.

Parameters

DisplayId

The id of the display device. Currently, only display id 0 (zero) is supported. For the **PB_EVENT_DRIVER_EXIT** event, only display id 0 is valid.

DisplayEvent

The display event that triggers the callback function. It may be one of the following:

PB_EVENT_DRIVER_UNDEFINED	Not currently used, but reserved for future use.
PB_EVENT_DRIVER_ERROR	Trigger when the xSides Driver encounters an error condition. An error code is returned. The error codes include:
XSD_FATAL_ERROR	A fatal error in the xSides Driver has been encountered.
XSD_CONTROL_NOT_FOUND	The video driver's Control() function cannot be found. This is a fatal error for the xSides software running on Windows 9x.
XSD_DRIVER_HOOK_FAILED	The attempt to hook the video driver for Driver Hook Overscan has failed.
PB_EVENT_SCREEN_SAVER_ON	Trigger when screen saver starts running.
PB_EVENT_SCREEN_SAVER_OFF	Trigger when screen saver stops running.

PB_EVENT_DOS_FULL_SCREEN_ON	Trigger when a DOS shell goes to full screen mode.
PB_EVENT_DOS_FULL_SCREEN_OFF	Trigger when a DOS shell ends its full screen mode.
PB_EVENT_DD_FULL_SCREEN_ON	Trigger when a DirectDraw window goes to full screen mode.
PB_EVENT_DD_FULL_SCREEN_OFF	Trigger when a DirectDraw window ends its full screen mode.
PB_EVENT_MONITOR_ON	Trigger when the display monitor is turned on.
PB_EVENT_MONITOR_OFF	Trigger when the display monitor is turned off, e.g. for power saver mode.
PB_EVENT_RESOLUTION_CHANGE	Trigger when the display resolution is changed, and return the new display resolution.
PB_EVENT_DD_FAILURE	Trigger when Direct Draw encounters an unrecoverable error.
PB_EVENT_MSG_MODE_BEGIN	Trigger when the message mode (aka "Blue Screen of Death") occurs.
PB_EVENT_MSG_MODE_END	Trigger when the message mode (aka "Blue Screen of Death") has ended.

fpCallback

The function to call when the specified display event occurs. The function returns a status and has one parameter that is a pointer to the display event data structure.

Returns

PB_SUCCESS	Function was successful.
PB_ERROR	An unrecoverable error occurred.

lpParam

lpParam is a pointer to the PBDisplayEvent data structure. The display id, the display event, and if applicable, new screen resolutions are contained in the structure.

Notes

Pixelbar_ClearDisplayEvent

UINT32 Pixelbar_ClearDisplayEvent (UINT32 DisplayId, UINT32 DisplayEvent)

Pixelbar_ClearDisplayEvent() clears a previously requested display event. When the display event set is cleared, the Pixel Bar software will no longer service the event.

- Returns one of these values:

PB_SUCCESS	Function was successful.
PB_ERROR	An unrecoverable error occurred.

Parameters

DisplayId

The id of the display device. Currently, only display id 0 (zero) is supported. For the **PB_EVENT_DRIVER_EXIT** event, only display id 0 is valid.

DisplayEvent

The display event being cleared, which was previously specified and associated with a callback function by Pixelbar_SetDisplayEvent(). It may be one of the following:

PB_EVENT_DRIVER_UNDEFINED	Not currently used, but reserved for future use.
PB_EVENT_DRIVER_ERROR	Trigger when the xSides Driver encounters an error condition. An error code is returned. The error codes include:
XSD_FATAL_ERROR	A fatal error in the xSides Driver has been encountered.
XSD_CONTROL_NOT_FOUND	The video driver's Control() function cannot be found. This is a fatal error for the xSides software running on Windows 9x.
XSD_DRIVER_HOOK_FAILED	The attempt to hook the video driver for Driver Hook Overscan has failed.

PB_EVENT_SCREEN_SAVER_ON	Trigger when screen saver starts running.
PB_EVENT_SCREEN_SAVER_OFF	Trigger when screen saver stops running.
PB_EVENT_DOS_FULL_SCREEN_ON	Trigger when a DOS shell goes to full screen mode.
PB_EVENT_DOS_FULL_SCREEN_OFF	Trigger when a DOS shell ends its full screen mode.
PB_EVENT_DD_FULL_SCREEN_ON	Trigger when a DirectDraw window goes to full screen mode.
PB_EVENT_DD_FULL_SCREEN_OFF	Trigger when a DirectDraw window ends its full screen mode.
PB_EVENT_MONITOR_ON	Trigger when the display monitor is turned on.
PB_EVENT_MONITOR_OFF	Trigger when the display monitor is turned off, e.g. for power saver mode.
PB_EVENT_RESOLUTION_CHANGE	Trigger when the display resolution is changed, and return the new display resolution.
PB_EVENT_DD_FAILURE	Trigger when Direct Draw encounters an unrecoverable error.
PB_EVENT_MSG_MODE_BEGIN	Trigger when the message mode (aka "Blue Screen of Death") occurs.
PB_EVENT_MSG_MODE_END	Trigger when the message mode (aka "Blue Screen of Death") has ended.

Notes

Pixelbar_EnumerateDisplayEvents

UINT32 Pixelbar_EnumerateDisplayEvents (UINT32 DisplayId, UINT32 *lpDisplayEventIdCount,UINT32 **lpDisplayEvents, UINT32 *lpDisplayEventsReturned, UINT32 *lpszData, UINT32 szOutputBuffer)

Pixelbar_EnumerateDisplayEvents() returns the number of display events and the list of display events that have been set for the specified display device.

- Returns one of these values:

PB_SUCCESS	Function was successful.
PB_ERROR	An unrecoverable error occurred.
PB_BUFFER_TOO_SMALL	The provided output buffer is too small to contain all the requested information. The information returned in the output buffer is not guaranteed to be coherent or valid. The function should be called again with an output buffer of the appropriate size.

Parameters

DisplayId

The id of the display device. Currently, only display id 0 (zero) is supported. For the **PB_EVENT_DRIVER_EXIT** event, only display id 0 is valid.

lpDisplayEventCount

The address of the UINT32 where the count of current display events will be returned. This value is equal to or greater than the value in *lpDisplayEventsReturned.*

lpDisplayEvents

The address of an array of UINT32 that hold the Display Events that are currently set for the display id.

lpDisplayEventsReturned

lpDisplayEventsReturned is the address of an UINT32 where the count of display events, which are returned in lpDisplayEvents, will be written. Unless the output buffer pointed to by lpDisplayEvents is too small; the value in lpDisplayEventsReturned is the same as in lpDisplayEventCount.

lpszData

The address of the size of the requested data. If the value is larger than szOutputBuffer, then the provided buffer is too small, and the error code PB_BUFFER_TOO_SMALL is returned by the function.

szOutputBuffer

The size of the output buffer pointed to by lpDisplayEvents, in bytes. This will allow the Pixel Bar software to refrain from overflowing the buffer.

Notes

Pixelbar_QuerySupport

UINT32 Pixelbar_QuerySupport (UINT32 DisplayId, UINT32 xSidesMode, UINT32 XRes, UINT32 YRes, UINT32 BitsPerPixel, UINT32 Edge, PBSize *lpPBSize, UINT32 *lpszData, UINT32 szOutputBuffer)

Pixelbar_QuerySupport() returns the maximum pixel bar size(s) for the specified xSides Mode, display resolution and display edge. A capability structure is returened.

- Returns one of these values:

PB_SUCCESS	Function was successful.
PB_ERROR	An unrecoverable error occurred.

Parameters

DisplayId

The id of the display device. Currently, only display id 0 (zero) is supported. For the **PB_EVENT_DRIVER_EXIT** event, only display id 0 is valid.

xSidesMode

The xSides mode that the pixel bar is to operate in. May be one of the following:

- MODE_SHARE — Share mode -- shares the current display resolution with the desktop by decreasing the desktop size.
- MODE_STEPUP — Step Up mode -- increases the physical resolution of the display, but keep the desktop at the current resolution and use the difference for displaying the pixel bar.
- MODE_STEPDOWN — Step Down mode -- maintains the physical resolution of the display, but force the desktop to the next smaller defined resolution and use the difference for displaying the pixel bar.
- MODE_OVERSCAN — Overscan mode -- reprograms the video hardware to provide extra scanlines at the bottom of the screen to display the pixel bar. This mode currently is valid only for pixel bars at the bottom or top edge of the display.

Note: The interaction of pixel bars running in different modes is currently undefined.

XRes

The horizontal resolution of the display.

YRes

The vertical resolution of the display.

BitsPerPixel

The number of bits per pixel.

Edge

Edge specifies which edge of the display surface the pixel bar will be displayed on. May be one or more of the following:

PB_TOP_EDGE	The top edge.
PB_BOTTOM_EDGE	The bottom edge.
PB_RIGHT_EDGE	The right edge. This edge is not valid if the pixel bar is in Qverscan mode.
PB_LEFT_EDGE	The left edge. This edge is not valid if the pixel bar is in Overscan mode.

lpPBSize

The address of a PBSize structure that holds the size of the maximum pixel bar size. Values of 0 in either the width or height fields indicate that the xSides Driver cannot support a pixel bar..

lpszData

The address of the size of the requested data. If the value is larger than szOutputBuffer, then the provided buffer is too small, and the error code PB_BUFFER_TOO_SMALL is returned by the function.

szOutputBuffer

The size of the output buffer pointed to by lpPBSize, in bytes. This will allow the Pixel Bar software to refrain from overflowing the buffer.

Notes

Pixelbar_EnumerateFileInformation

UINT32 Pixelbar_EnumerateFileInformation (UINT32 *lpTotalFileCount, PBFileInfo *lpFileInfo, UINT32 *lpNumFilesReturned, UINT32 *lpDataSizeReturned, UINT32 *lpszData, UINT32 szOutputBuffer)

Pixelbar_EnumerateFileInformation() returns the names and versions of the currently installed xSides driver files.

- Returns one of these values:

PB_SUCCESS	Function was successful.

PB_ERROR	An unrecoverable error occurred.

Parameters

lpTotalFileCount

Pointer to the total number of file entries.

lpFileInfo

Address of the buffer where the file information is to be placed.

lpNumFilesReturned

Pointer to the number of returned file entries.

lpDataSizeReturned

Address where the amount of data -- in number of bytes -- that is returned (in the buffer specified by lpFileinfo) is placed.

lpszData

Address where the total amount of data -- in number of bytes -- available for the file information is placed.

szOutputBuffer

Size of the buffer pointed to by lpFileVer.

Notes

Data Definitions

The data types used in the Pixel Bar API are defined here. These definitions assume a Little Endian (aka Intel) byte ordering where the most significant byte has the highest address. For the structures depicted below, the most significant byte is on the left, and the least significant byte is on the right, with each line depicting 32 bits (4 bytes).

Integer Types

UINT8	Unsigned 8 bit integer.
UINT16	Unsigned 16 bit integer.
UINT32	Unsigned 32 bit integer.
UINT64	Unsigned 64 bit integer.
INT8	Signed 8 bit integer.

INT16	Signed 16 bit integer.
INT32	Signed 32 bit integer.
INT64	Signed 64 bit integer.

Macros

Return Codes:

PB_SUCCESS	0
PB_ERROR	1
PB_DISPLAY_ID_OUT_OF_BOUNDS	2
PB_DISPLAY_NOT_INITIALIZED	3
PB_INVALID_EDGE_FOR_MODE_OVERSCAN	4
PB_DISPLAY_EDGE_ALREADY_HAS_PIXELBAR	5
PB_PIXELBAR_ID_OUT_OF_BOUNDS	6
PB_PIXELBAR_NOT_CREATED	7
PB_INVALID_EDGE_FOR_MODE_SHARE	8
PB_INVALID_MODE	9
PB_INVALIDATED_ID	10
PB_BAD_PARAMETERS	11
PB_ACCVIO	12
PB_DISPLAY_BUSY	13
PB_UNSUPPORTED_RESOLUTION	14
PB_XSIDES_NOT_INSTALLED	15
PB_SUCCESS_NO_RES_CHANGE	16
PB_BUFFER_TOO_SMALL	17
PB_INSUFFICIENT_RESOURCES	18
PB_ALREADY_DONE	19
PB_PIXELBAR_DEACTIVE	20
PB_DISPLAY_REGION_OUT_OF_BOUNDS	21
PB_VERSION_MISMATCH	22
PB_XSIDES_SUPPORTED	23

xSides Modes:

MODE_OVERSCAN	0	
MODE_SHARE	1	
MODE_STEPUP	2	
MODE_STEPDOWN	3	
MODE_AUTO	4	// Overscan if possible, otherwise Share.

Status Flags: // Why aren't these called *_Masks?

PB_STATUS_ACTIVE	1	// Bit 0
PB_STATUS_EXACT_SIZE	2	// Bit 1
PB_STATUS_INVALID	4	// What is this used for?
PB_STATUS_VALID_RES	8	
PB_STATUS_MOUSE_ENABLED	16	// What is this?
PB_STATUS_DISPLAYED	9	// What is this?

Display Edges: // Change from bitmasks to ordinals.

PB_BOTTOM_EDGE	0
PB_RIGHT_EDGE	1
PB_TOP_EDGE	2
PB_LEFT_EDGE	3

Display Edge Masks:

PB_BOTTOM_EDGE_MASK	1
PB_RIGHT_EDGE_MASK	2
PB_TOP_EDGE_MASK	4
PB_LEFT_EDGE_MASK	8

Mouse Events:

PB_MOUSE_LBUTTON_DOWN	1
PB_MOUSE_LBUTTON_UP	2
PB_MOUSE_RBUTTON_DOWN	4
PB_MOUSE_RBUTTON_UP	8

PB_MOUSE_MBUTTON_DOWN	16
PB_MOUSE_MBUTTON_UP	32
PB_MOUSE_LDBL_CLICK	64
PB_MOUSE_RDBL_CLICK	128
PB_MOUSE_MDBL_CLICK	256
PB_MOUSE_MOVEMENT	512
PB_MOUSE_LEAVE	1024

Display Events:

PB_EVENT_DRIVER_UNDEFINED	0
PB_EVENT_DRIVER_ERROR	1
PB_EVENT_SCREEN_SAVER_ON	2
PB_EVENT_SCREEN_SAVER_OFF	3
PB_EVENT_DOS_FULL_SCREEN_ON	4
PB_EVENT_DOS_FULL_SCREEN_OFF	5
PB_EVENT_DD_FULL_SCREEN_ON	6
PB_EVENT_DD_FULL_SCREEN_OFF	7
PB_EVENT_MONITOR_ON	8
PB_EVENT_MONITOR_OFF	9
PB_EVENT_RESOLUTION_CHANGE	10
PB_EVENT_DD_FAILURE	11
PB_EVENT_MSG_MODE_BEGIN	12
PB_EVENT_MSG_MODE_END	13

xSides Driver Error

XSD_FATAL_ERROR	0
XSD_CONTROL_NOT_FOUND	1
XSD_DRIVER_HOOK_FAILED	2

XSD_FILE_TYPE:

PB_DLL	10
PB_XSIDES_DLL	11
PB_XSIDESK_SYS	12
PB_XSD9XDLL_DLL	13
PB_XSIDES_VXD	14

Structure Types

PBConfiguration

UINT16 Version	UINT16 Size
UINT32 DisplayId	
UINT32 PixelBarMask	
UINT32 MaxOverscanWidth	
UINT32 MaxOverscanHeight	
UINT32 MaxOverscanBPP	
UINT32 EdgeFlags	

Version

The structure version. This occupies the third and fourth bytes of the structure (bytes 2 and 3).

Size

The structure size, in bytes, including the version and size fields. This occupies the first two bytes of the structure (bytes 0 and 1).

DisplayId

The display to apply the configuration information.

PixelBarMask

A bitmask indicating the pixel bars that will be displayed. A union of the pixel bar edges that will be used during the current xSides session.

MaxOverscanWidth

The maximum cumulative width, in pixels, of the overscan regions. A value of 0 indicates that overscan will not be used.

MaxOverscanHeight

The maximum cumulative height, in pixels, of the overscan regions. A value of 0 indicates that overscan will not be used.

MaxOverscanBPP

The maximum pixel resolution that overscan pixel bars will be displayed in. Can be 8, 16, 24 or 32. A value of 0 indicates that overscan mode will not be used.

EdgeFlags

A bitfield indicating which of the 4 sides will have pixel bars. A union of 1 or more of the following:

PB_BOTTOM_EDGE

PB_RIGHT_EDGE

PB_TOP_EDGE

PB_LEFT_EDGE

PBDisplayInfo

UINT16 Version	UINT16 Size
UINT32 DisplayId	
UINT32 HwRezX	
UINT32 HwRezY	
UINT32 TotalRezX	
UINT32 TotalRezY	
UINT32 DesktopRezX	
UINT32 DesktopRezY	
UINT32 BitsPerPixel	

UINT32 RefreshFrequency
UINT32 InterlaceFlag
UINT32 Stride

Version

The structure version. This occupies the third and fourth bytes of the structure (bytes 2 and 3).

Size

The structure size, in bytes, including the version and size fields. This occupies the first two bytes of the structure (bytes 0 and 1).

DisplayId

The display that the information is being returned for.

HwRezX

The number of pixels along the X axis that are being displayed by the video driver. This number doesn't include any overscan regions.

HwRezY

The number of pixels along the Y axis that are being displayed by the video driver. This number doesn't include any overscan regions.

TotalRezX

The number of pixels along the X axis that are being displayed by the video hardware, including all overscans regions.

TotalRezY

The number of pixels along the Y axis that are being displayed by the video hardware, including all overscan regions.

DesktopRezX

The number of pixels along the X axis that are being used by the desktop. This excludes all regions used by active pixel bars.

DesktopRezY

The number of pixels along the Y axis that are being used by the desktop. This excludes all regions used by active pixel bars.

BitsPerPixel

The current pixel resolution of the display, expressed as the number of bits per pixel. It is assumed that the display is a packed flat, non-banked frame buffer.

RefreshFrequency

The current refresh frequency of the display.

InterlaceFlag

TRUE (1) if the display is interlaced, otherwise FALSE (0).

Stride

The number of bytes between adjacent scan lines in the video frame buffer, including the bytes in the displayed scan line. If not known, then it should be filled with 0 (zero).

PBPixelbarState

UINT16 Version	UINT16 Size
UINT32 PixelbarId	
UINT32 DisplayId	
UINT32 XSidesMode	
UINT32 Status	
UINT32 Edge	
UINT32 OriginX	
UINT32 OriginY	
UINT32 Width	
UINT32 Height	

Version

The structure version. This occupies the third and fourth bytes of the structure (bytes 2 and 3).

Size

The structure size, in bytes, including the version and size fields. This occupies the first two bytes of the structure (bytes 0 and 1).

PixelbarId

The Id of the pixel bar.

DisplayId

The Id of the display that the pixel bar is on.

xSidesMode

The mode that the pixel bar is in. May be one of the following:

MODE_OVERSCAN

MODE_SHARE

MODE_STEPUP

MODE_STEPDOWN

MODE_AUTO

Status

The current status of the pixel bar. The following bits are defined and may be set to TRUE (1) or FALSE (0).

PB_STATUS_ACTIVE

PB_STATUS_EXACT_SIZE

Edge

The edge of the display that the pixel bar is located at. May be one of the following:

OriginX

The X origin of the pixel bar, in hardware screen coordinates.

OriginY

The Y origin of the pixel bar, in hardware screen coordinates.

Width

The current width of the pixel bar, in pixels.

Height

The current height of the pixel bar, in pixels.

CURSOR_POS

UINT32 X_Position
UINT32 Y_Position

X_Position

The X_Position of the cursor.

Y_Position

The Y_Position of the cursor.

PBMouseEvent

UINT16 Version	UINT16 Size
UINT32 PixelbarId	
UINT32 MouseEvent	
UINT32 ButtonState CURSOR_POS CursorPos UINT32 RepeatCount	

Version

The structure version. This occupies the third and fourth bytes of the structure (bytes 2 and 3).

Size

The structure size, in bytes, including the version and size fields. This occupies the first two bytes of the structure (bytes 0 and 1).

PixelbarId

Indicates which pixel bar the mouse event occurred in.

MouseEvent

Indicates the mouse event being reported. May be one of:

- PB_MOUSE_LBUTTON_DOWN
- PB_MOUSE_LBUTTON_UP

- PB_MOUSE_RBUTTON_DOWN
- PB_MOUSE_RBUTTON_UP
- PB_MOUSE_MBUTTON_DOWN
- PB_MOUSE_MBUTTON_UP
- PB_MOUSE_LDBL_CLICK
- PB_MOUSE_RDBL_CLICK
- PB_MOUSE_MDBL_CLICK
- PB_MOUSE_MOVEMENT
- PB_MOUSE_LEAVE

ButtonState

What is this?

CursorPos

The position of the cursor relative to the pixel bar origin of the mouse movement. Valid only if the mouse event is PB_MOUSE_MOVEMENT.

RepeatCount

What is this?

PBMouseEventInfo

UINT16 Version	UINT16 Size
UINT32 PixelbarId	
UINT32 MouseEventId	
UINT32 MouseEvent	
UINT32 fpCallbackFunction	

Version

The structure version. This occupies the third and fourth bytes of the structure (bytes 2 and 3).

Size

The structure size, in bytes, including the version and size fields. This occupies the first two bytes of the structure (bytes 0 and 1).

PixelbarId

The Pixel Bar that the Mouse Event set is associated with.

MouseEventId

The Id of the Mouse Event set.

MouseEventId

Indicates the mouse event being reported. May be one or more of:

- PB_MOUSE_LBUTTON_DOWN
- PB_MOUSE_LBUTTON_UP
- PB_MOUSE_RBUTTON_DOWN
- PB_MOUSE_RBUTTON_UP
- PB_MOUSE_MBUTTON_DOWN
- PB_MOUSE_MBUTTON_UP
- PB_MOUSE_LDBL_CLICK
- PB_MOUSE_RDBL_CLICK
- PB_MOUSE_MDBL_CLICK
- PB_MOUSE_MOVEMENT
- PB_MOUSE_LEAVE

fpCallbackFunction

The address of the callback function associated with this Mouse Event set.

PBDisplayEvent

UINT16 Version	UINT16 Size
UINT32 DisplayId	
UINT32 DisplayEvent	
UINT32 SubCode	
PBDisplayInfo NewDisplayInfo | UINT32 ErrorCode	

Version

The structure version. This occupies the third and fourth bytes of the structure (bytes 2 and 3).

Size

The structure size, in bytes, including the version and size fields. This occupies the first two bytes of the structure (bytes 0 and 1).

DisplayId

Indicates which display device the display event occurred in.

DisplayEvent

Indicates the display event being reported. May be one of the following:

- PB_EVENT_DRIVER_ERROR
- PB_EVENT_SCREEN_SAVER_ON
- PB_EVENT_SCREEN_SAVER_OFF
- PB_EVENT_DOS_FULL_SCREEN_ON
- PB_EVENT_DOS_FULL_SCREEN_OFF
- PB_EVENT_DD_FULL_SCREEN_ON
- PB_EVENT_DD_FULL_SCREEN_OFF
- PB_EVENT_MONITOR_ON
- PB_EVENT_MONITOR_OFF
- PB_EVENT_RESOLUTION_CHANGE
- PB_EVENT_DD_FAILURE
- PB_EVENT_MSG_MODE_BEGIN

- PB_EVENT_MSG_MODE_END

SubCode

Indicates a sub event code for the Display Event. The default value is 0.

For PB_EVENT_DRIVER_ERROR, the valid subcodes are:

- XSD_FATAL_ERROR
- XSD_CONTROL_NOT_FOUND
- XSD_DRIVER_HOOK_FAILED

NewDisplayInfo

Data reflecting the new state of the display when the event is PB_EVENT_RESOLUTION_CHANGE.

PBPixelbarSize

UINT16 Version	UINT16 Size
UINT32 BaseWidth	
UINT32 BaseHeight	
INT32 OriginX	
INT32 OriginY	
UINT32 Width	
UINT32 Height	

Version

The structure version. This occupies the third and fourth bytes of the structure (bytes 2 and 3).

Size

The structure size, in bytes, including the version and size fields. This occupies the first two bytes of the structure (bytes 0 and 1).

BaseWidth

BaseWidth specifies the horizontal number of pixels of the resolution that the pixel bar size applies to. For example, if the size of the pixel bar for the resolution of 640x480 is being specified, then the BaseWidth would be set to 640.

BaseHeight

BaseHeight specifies the vertical number of pixels of the resolution that the pixel bar size applies to. For example, if the size of the pixel bar for the resolution of 640x480 is being specified, then the BaseWidth would be set to 480.

OriginX

OriginX specifies the x position on the display surface, in units of pixels with respect to the Desktop Origin, where the origin of the pixel bar (which is the upper left corner of the pixel bar) is located. The pixel bar origin must be consistent with its screen edge parameter; otherwise an error will be returned. This is a signed value and may be a negative value (which it will be for non-zero width pixel bars on the left edge of the screen.)

OriginY

OriginY specifies the y position on the display surface, in units of pixels with respect to the Desktop Origin, where the origin of the pixel bar (which is the upper left corner of the pixel bar) is located. The pixel bar origin must be consistent with its screen edge parameter; otherwise an error will be returned. This is a signed value and may be a negative value (which it will be for non-zero height pixel bars on the top edge of the screen.)

Width

Width specifies the desired width of the pixel bar, in number of pixels. It may not be larger than the total physical horizontal resolution of the display, which include the desktop and the pixel bars.

Height

Height specifies the desired height of the pixel bar, in number of pixels. It may not be larger than the total physical vertical resolution of the display, which include the desktop and the pixel bars.

PBSize

UINT16 Version	UINT16 Size
UINT32 Width	

UINT32 Height

Version

The structure version. This occupies the third and fourth bytes of the structure (bytes 2 and 3).

Size

The structure size, in bytes, including the version and size fields. This occupies the first two bytes of the structure (bytes 0 and 1).

Width

Width specifies the desired width of the pixel bar, in number of pixels. It may not be larger than the physical horizontal resolution of the display.

Height

Height specifies the desired height of the pixel bar, in number of pixels. It may not be larger than the physical vertical resolution of the display.

PBFileInfo

UINT16 Version	UINT16 Size
XSD_FILE_TYPE edFileType	
UINT32 szFileSpec	
UINT32 szFileVersion	
UINT32 dwFileSpecIndex	
UINT32 dwFileVersionIndex	
char FileData[1]	

Version

The structure version. This occupies the third and fourth bytes of the structure (bytes 2 and 3).

Size

The structure size, in bytes, including the version and size fields. This occupies the first two bytes of the structure (bytes 0 and 1). Since this is a variable size structure, the size field allows the a buffer of PBFileInfo structures to be parsed.

edFileType

Width specifies the desired width of the pixel bar, in number of pixels. It may not be larger than the physical horizontal resolution of the display.

szFileSpec

The number of bytes in the FileSpec, which is a null terminated character string, including the directory path and file name of the xSides driver file. The null terminator is not counted.

szFileVersion

The number of bytes in the FileVersion, which is a null terminated character string, including version information for the xSides driver file. The null terminator is not counted.

dwFileSpecIndex

Index into the.

dwFileVersionIndex

Height specifies the desired height of the pixel bar, in number of pixels. It may not be larger than the physical vertical resolution of the display.

FileData

A character array where the FileSpec and FileVersion information are stored as null terminated strings.

Index

Pixel Bar Header File